Title
Tree LANS with collision avoidance : protocol, switch architecture, and performance

Authors
Suda, Tatsuya
Morris, Steve
Goto, Kunio

Publication Date
1987

Peer reviewed

Tree LANs with Collision Avoidance: Protocol, Switch Architecture and Performance*

Tatsuya Suda, Steve Morris, Kunio Goto
Department of Information and Computer Science
University of California, Irvine
Irvine, CA 92717, U.S.A
(phone) 714-856-5474

Technical Report 87-17

Abstract

Packet collisions and their resolution create a performance bottleneck in random access LANs. A hardware solution to this problem is to use collision avoidance switches. These switches allow the implementation of random access protocols without the penalty of collisions among packets. We review and compare the designs of some tree LANs that use collision avoidance switches. They have the potential of combining the benefits of random access (low delay when traffic is light, simple and distributed, and therefore robust, protocols) with excellent network utilization and concurrency of transmission. The collision avoidance LANs we review are broadcast star, Hubnet-like tree, Tinker-Tree, and a treenet that allows concurrent broadcasts within non-intersecting subtrees. After this review, we present a slotted-time, infinite user analysis of the broadcast star network.

* This material is based upon work supported by the National Science Foundation under Grant No. DCI-8602052. This research is also in part supported by the University of California MICRO program and by the University of California, Irvine, the Academic Senate Committee on Research.

1. Introduction

A little over a decade ago, the first distributed algorithms, which allow a number of geographically separated stations to communicate over a single shared communications channel, were implemented in the ALOHA network. Since then, the use of these and similar protocols, known as multiple access protocols, has spread into hundreds of networks.

Multiple access protocols divide broadly into two classes: random (or contention) access protocols and controlled access protocols [1]. In random access protocols, transmission rights are simultaneously offered to a group of stations in the hope that exactly one of the stations has a packet to send. If, however, two or more stations send packets simultaneously on the channel, these messages interfere with each other and none of them are correctly received by the destination stations. There is said to be a *collision* among the packets. In such cases, stations retransmit packets until they are successfully received by the destination stations.

Controlled access protocols avoid collisions by coordinating access of the stations to the channel by imposing either a predetermined or dynamically determined order of access. Coordination among stations about sharing the channel is done by use of the channel itself. Each station indicates with a short message on the channel whether or not it wants to access the channel. This polling mechanism consumes some channel capacity for each station, even if it does not require access to the channel. While such protocols are efficient when traffic is heavy, under light traffic conditions, they result in unnecessary packet delays as stations that want to transmit wait their turn.

Random access protocols exhibit small packet delays under light traffic conditions: stations transmit as soon as they want access to the channel, and the probability of a collision is low when traffic is light. Another attractive aspect of random access protocols is their simplicity, making them easy to implement at stations. Also, some or all of the protocol can be distributed to the stations. This promotes robustness because there is less or no exposure to the failure of a few master nodes. On the other hand, random access protocols have a performance bottleneck under heavy traffic conditions because of large numbers of collisions and the time required to resolve them through retransmissions.

Most random access protocols handle the resolution of collisions by using some channel capacity to allow contending stations to establish a schedule of transmissions among themselves. Although individual resolution protocols vary in their details, a large portion of them use the following general scheme. For any particular collision there is a set of contending stations. The resolution algorithm divides this set of stations into several smaller groups of stations, and each group in turn is allowed to access the channel. If a group consists of more than one station, then there will be collisions, and the algorithm is applied to this new, smaller group of contenders. A station is able to transmit without collision when it ends up in a group comprising just that station. The algorithm recursively divides stations into smaller and smaller groups. Examples of such protocols are: urn, tree resolution, and CSMA/CD.

These protocols use collisions of packets as the means of providing information to each

station on the state of the algorithm. There is therefore an unavoidable loss of channel capacity with these protocols. As the traffic intensity in such a network increases, so do the chances for collisions, and channel utilization and packet delay suffer accordingly. It would be nice to have a protocol which has the benefits of random access, in particular low delay in light traffic, but which also does not suffer from lost channel capacity when traffic is heavier. Protocols that uses collision avoidance switches are such protocols. Before discussing examples of these, we present CSMA/CD as a more detailed example of a random access protocol. We choose CSMA/CD because it is closest in character to the collision avoidance LANs that we discuss in this paper.

2. CSMA/CD LANs

In CSMA/CD, stations share a single broadcast channel. When a station has a packet to send, it senses the state of the channel. If the channel is busy, the station waits an amount of time prescribed by a deferral algorithm, and then senses the state of the channel again. This process is repeated until the station senses the channel as being idle, at which time the station transmits its packet.

If two or more stations, having sensed the channel as being idle, transmit their packets at, or about the same time, their transmissions will interfere with each other resulting in a collision on the channel. Each transmitting station terminates its transmission as soon as it detects a collision and then waits an amount of time prescribed by a resolution algorithm before attempting to retransmit its packet. CSMA/CD resolution algorithms use some form of randomization to vary the times that stations wait before retransmitting. Otherwise contending stations will recollide indefinitely. Nevertheless, two or more stations can randomly choose the same time to retransmit. If this occurs, each of the colliding stations reapplies the resolution algorithm. This process continues until all stations have transmitted successfully.

In CSMA/CD, when traffic is light there is a good chance that the initial transmission of a station will succeed. This random access characteristic avoids the wasted channel capacity and packet delays that occur with fixed assignment and reservation assignment protocols under light traffic conditions. Under moderate to heavy traffic, the probability of collisions in CSMA/CD increases, and channel utilization decreases because of the transmission of collided packet fragments and because of the time required to resolve contention.

3. Collision Avoidance

A different approach to the collision problem has been developed by Closs and R. Lee [2], Albanese [3], Boulton and P. Lee [4], and Suda, Yemini, and Schwartz [5]. This approach uses hardware called collision avoidance switches to prevent collisions by arbitrating random access to a communications channel. While the channel is being used by one station, other stations are blocked from using it. Networks based on these switches can provide low packet delay under light load and simple access protocols, and not waste channel utilization due to collision resolution or the transmission of collided packets. We will give some specifics of three LANs which have tree topologies and use collision avoidance switches. The simplest such LAN is a broadcast star.

4. Collision Avoidance Broadcast Star

In this network stations are connected by full duplex channels to a central switch. Each of these channels comprise an uplink and a downlink. See Figure 1. The switch may be viewed functionally as containing two components: the *selector* and the *broadcaster*. The selector selects one packet from the uplinks and transmits this packet on a single output line to the broadcaster. The broadcaster receives the packet from the selector's output line and retransmits it on all the downlinks. The selector has two states. It is *busy* from the time it has selected a packet to the time is has finished transmitting the packet to the broadcaster. Otherwise the selector is *idle*. While the selector is busy, all packets arriving on uplinks are ignored in their entirety. Upon going idle, the selector selects the next newly arriving packet.

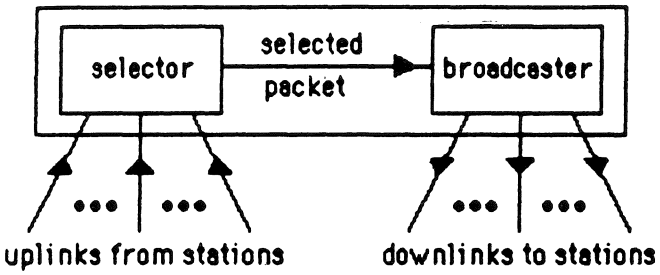

Fig.1 Broadcast Star Switch

Although it is the selector component alone that implements the collision avoidance, we shall refer to the whole switch as a collision avoidance switch. The important feature of a collision avoidance switch is that when two or more packets contend for the output line, it is guaranteed that one of the packets acquires the line and is successfully transmitted on it. Thus, no channel time is wasted in the transmission of collided packets, and the traditional penalty of random access is eliminated. A simple, pure random access protocol can be used without the need for a contention resolution subprotocol. Collision avoidance can be implemented with very little circuitry. An implementation example is given in [3]. It should be noted that these switches do not buffer packets; no memory is used to store-and-forward packets.

The station protocol for the broadcast star is very simple and is like pure ALOHA :

(1) A station transmits a packet as soon as one is available.

(2) After a propagation delay to and from the switch, the station monitors its downlink for the start of its packet.

(3) If the station does not see the start of its packet, then it retransmits the packet immediately,

(4) else the station does see its packet and knows that the packet has *won* the switch and will be broadcast in its entirety.

In ALOHA, in order to avoid endless recollisions, each station waits a random time after a

collision before retransmitting. In the above protocol, stations do not defer retransmission. This lack of deferment does not hurt performance because no collision with an ongoing transmission can occur in this broadcast star network. Because contending stations can retransmit failed packets without delaying, the switch idle-time between successively selected packets can become small. Because the output channel is used solely for data transmission, its utilization is limited chiefly by the rate at which blocked packets are resubmitted to the switch. Stations resubmit their packets as soon as they learn of transmission failure. Thus, resubmission rates depend solely on the round trip propagation time between stations and the switch. Later in this paper we shall present analysis that shows that a collision avoidance broadcast star network can obtain channel utilization very close to 1.

4.1. Comparison of Broadcast Star to CSMA/CD

Let T be the worst one way propagation delay between two stations in a CSMA/CD network. Suppose station X starts transmitting at time t. Then by time $t+T$ all stations will have seen the beginning of X's packet. X's packet will experience a collision only if another station starts transmitting before sensing it. Thus during the time t to $t+T$, X's packet is vulnerable to collision. If no collision occurs during this period then none will occur for the duration of the transmission, because all stations will have sensed the packet and therefore refrain from transmitting. In this case station X may be said to have seized the channel. However, station X can not conclude that it has seized the channel until it has seen no collision during the longer period t to $t+2T$. To see this, imagine that stations X and Y are separated by the propagation delay of T. Suppose X transmits at time t and Y transmits at time $t+T-\epsilon$, where ϵ is arbitrarily small. A collision of the two packets occurs, but X will not see the collision until Y's packet reaches X at time $t+2T-\epsilon$. This $2T$ period of time before a station knows it has seized the channel may be called the *collision window.*

One may think of the broadcast star as behaving like a CSMA/CD network with a collision window of length zero. In this analogy, the sending side of each CSMA/CD station comprises a broadcast star station, its uplink, and part of the selector of the central switch. (The uplink is like a very long wire within the CSMA/CD station!) The selector functions as the channel sensing logic for each station. The receiving side of each CSMA/CD station comprises a broadcast star station, its downlink (another long wire), and part of the broadcaster of the central switch. The CSMA/CD channel is the wire connecting the selector and the broadcaster. This wire can be considered as being arbitrarily short.

In this analogy, a packet enters the channel when it is passed by the selector to the broadcaster. Because collisions never occur in this CSMA/CD LAN, the analogy is that every other station senses a packet on the channel as soon as it enters the channel, and the collision window is of length zero.

In CSMA/CD, a station must transmit for the duration of the collision window before knowing that it has seized the channel and that its packet will not suffer a collision. Therefore the station protocol requires that packets have a minimum duration equal to the length of the collision window. This length is a fixed value that depends on the greatest distance between two stations. On the other hand, packet duration is inversely proportional to channel speed.

Doubling the channel speed in a CSMA/CD network requires doubling the number of bits in the minimum packet length. This creates a conflict that limits the maximum channel speed that can be utilized in CSMA/CD, short of padding packets with bits to meet the minimum packet size. Thus CSMA/CD is not a good protocol for high speed networks.

A substantial advantage of the broadcast star is that there is no minimum packet length. This makes it suitable for high speed networks such as those using optical fiber technology. Also note that, because communication to and from the star switch is done over point-to-point links, its architecture is compatible with standard fiber optics technology.

5. CASB Trees

LANs similar to broadcast stars but having a general rooted tree topology have been proposed and built [2, 4]. These networks have been called *hierarchical stars* and *hub networks*. In this paper we refer to them as CASB trees (Collision Avoidance, Single Broadcast trees). The nodes of the CASB tree consist of identical switches. Each switch is connected to its parent and children by full duplex channels consisting of an uplink and a downlink. The stations of the network are connected to the leaf switches of the tree. A child of a switch can be another switch or it can be a station. The root switch has no parent; its "parent" uplink is connected directly to its "parent" downlink. See Figure 2.

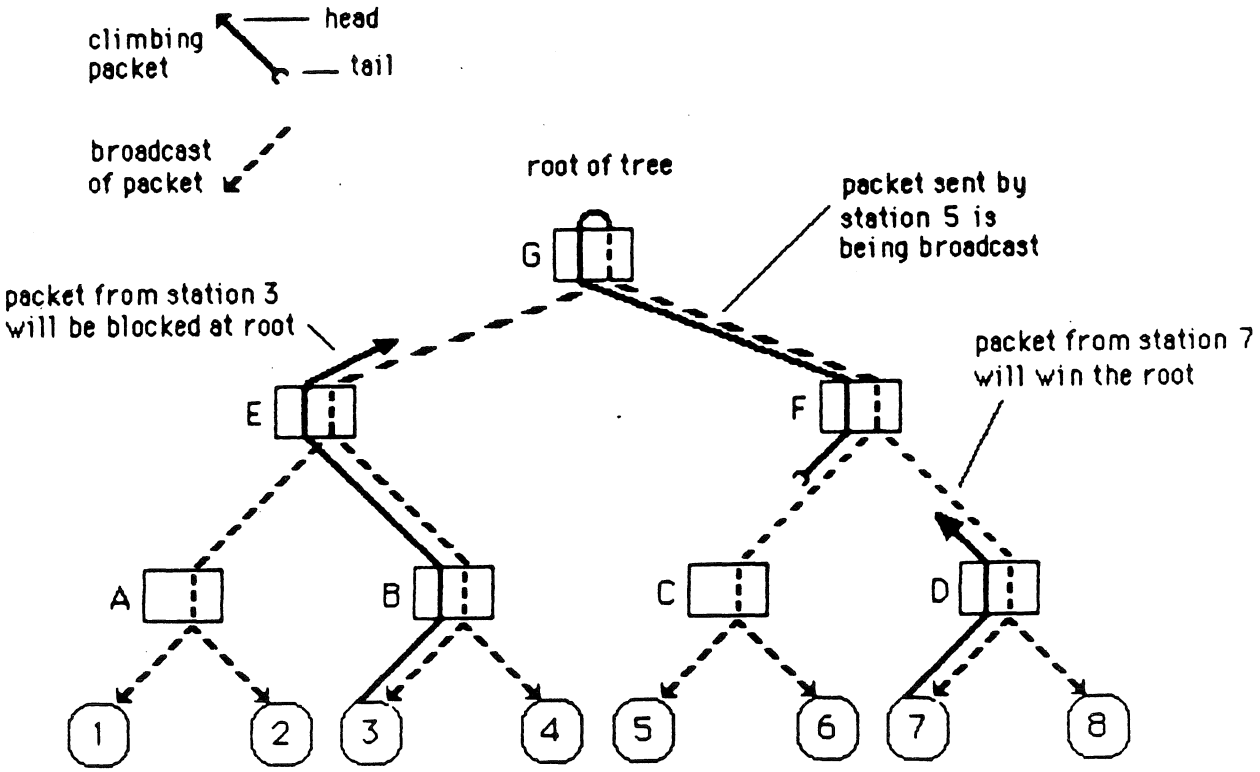

Fig.2 Transmissions in CASB Tree Network

The switch architecture is similar to the broadcast star switch; it contains a selector and a broadcaster. However these two components are not linked. Instead the selector passes its output to its parent switch, and the broadcaster receives its input from its parent switch. See Figure 3.

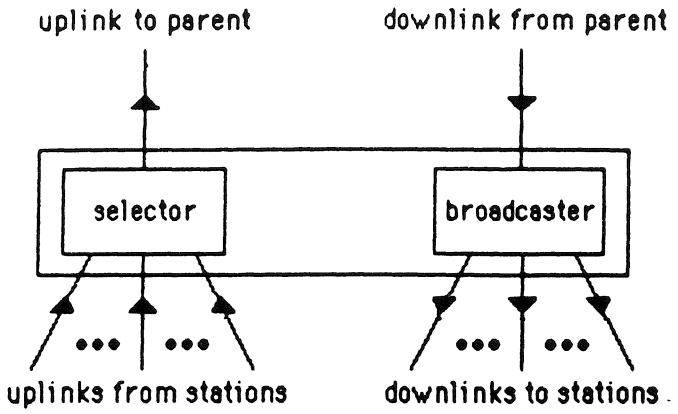

Fig.3 CASB Switch

Stations in the CASB network use the same protocol as in the simple broadcast star network. A station transmits as soon as a packet is ready and then monitors its downlink for the broadcast of its packet. The packet *climbs* the tree by being selected by each switch along the path to the root. The root switch serves the same function as the central switch in the broadcast star. Any packet selected by the root's selector is broadcasted to the children of the root. These children repeat the broadcast to their children, and so forth until the broadcast reaches every station. Once a packet is selected by the root, it is assured of being broadcast in its entirety to the whole tree. When a station sees the beginning of the broadcast of its packet, the station knows the transmission will not fail because of contention. However, if the station does not see the beginning of its packet within a round trip propagation to the root and back, it knows that the packet was blocked at a selector somewhere in the tree. The station then retransmits its packet immediately.

The CASB network functions like a broadcast star. The root node corresponds to a central switch. The selection of packets is distributed over several levels of switches, as is the propagation of packet broadcasts. Three factors would lead one to use a CASB tree instead of a simple broadcast star.

(1) The central switch of a broadcast star can accommodate only a limited number of stations; CASB trees allow for easy expansion when the fanout of a switch is fully occupied. Because all switches in the network are identical, expansion entails simply connecting a new switch as a the child of the full switch.

(2) The cabling cost of a broadcast star may be excessive. The intermediate switches in the CASB allow cables to be shared instead running a separate cable for each station to a central switch. This may be particularly attractive if some cable runs are long or are

difficult to install.

(3) CASB provides some fault tolerance that is not present in the broadcast star, because one switch failure does not disable the whole network. On the other hand, the increased number of connections in a CASB network increases fault exposure.

6. CAMB Tree

A tree network with collision avoidance which supports concurrent transmissions has been proposed by Suda, Yemini, and Schwartz [5]. In this paper we call this network a CAMB tree (Collision Avoidance Multiple Broadcast tree). The CAMB tree is a CASB tree with switch modifications that allow any switch to broadcast to the subtree below it. A packet reaches its destination by *climbing* the tree and being broadcast by its *proper ancestor*. The proper ancestor of a packet is the switch that roots the minimal subtree containing both the source and destination stations of the packet.

6.1. CAMB Switch Architecture

The CAMB switch has three components: the Uplink Selector (US), the Address Recognizer (AR), and the Downlink Selector (DS). See Figure 4. The US is identical to the selector of a CASB switch. The only difference is that it sends a selected packet to the AR, not to the parent uplink as in the CASB switch. The AR determines if the switch is the proper ancestor of a selected packet. The DS is a slightly complicated version of the broadcaster in the CASB switch. We next describe the functioning of the AR and DS in more detail.

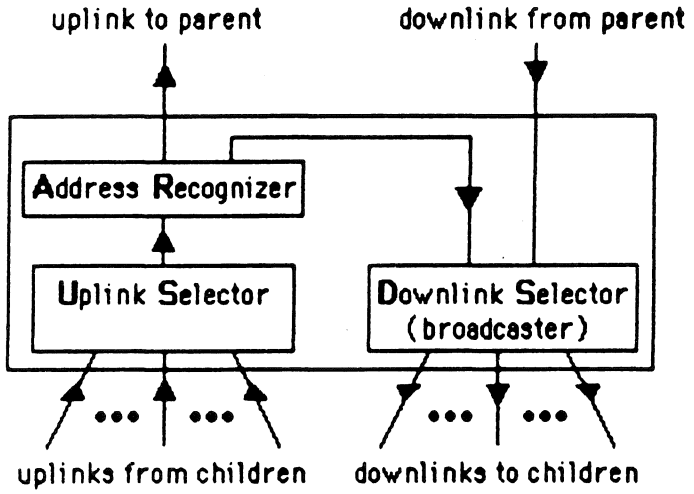

Fig.4 CAMB Switch

The AR knows the unique address of its switch, and the source station of a packet places in its header the address of the proper ancestor for that packet. The AR uses the following protocol:

(1) The AR checks the header of each packet passed to it by the US to see if the switch is the proper ancestor.

(2) If the switch is not the proper ancestor, the AR transmits the packet to the parent uplink,

(3) else the switch is the proper ancestor, and the AR checks the status of the DS,

(3-1) if the DS is busy broadcasting a packet from the parent downlink, then the AR discards the packet it is receiving. (This event is called a packet *preemption*).

(3-2) else the DS is idle, and the AR transmits the packet to the DS and to the parent uplink. (We will explain in Section 6.3 why the packet is also passed to the parent uplink even though it has been selected by its proper ancestor for broadcast).

The DS receives packets from either the AR or the parent downlink. It broadcasts each packet it receives by repeating the packet on all the children downlinks. The DS uses the following protocol:

(1) If the DS is idle when it starts to receive a packet from either the AR or the parent downlink, then it will start to broadcast that packet.

(2) If the DS is busy broadcasting a packet from the AR when it starts to receive a packet from the parent downlink, the AR-broadcast is terminated and the packet from the parent is broadcast instead. (This event is called a broadcast *abortion*).

Packets received from the parent downlink are given priority over packets received from the AR. If, instead, the switch blocked a packet received from the parent downlink, then the situation could arise where a source station received the broadcast of its packet but the destination station did not. This situation is incompatible with the station protocol, in which the source station uses the receipt of its packet as an indication that the packet was also broadcast to the destination station. The station protocol is described further in Section 6.2.

It should be noted that the switch does not store and forward packets, although small amounts of memory might be necessary for buffering in support of processing such as checking the address in a header. Further, the amount of logic required by the switch should be quite small and the processing performed by the switch quite simple. This may be useful as link channel speeds increase to the G bits/sec level and beyond through the use of optical transmission lines. By keeping the switch logic relatively simple in terms of the number of devices used, it may be economically feasible for this architecture to take advantage of emergent technologies such as photonic devices. If conventional electronic logic is used inside the switch, a bottleneck may occur: the conversion of light signals to electronic signals, their processing by conventional electronic logic, and their conversion back to light signals, may limit the transmission rate.

6.2. Station Protocol

The CAMB station protocol is based upon the station monitoring its downlink for the broadcast of its packet. In the broadcast star, or in the CASB tree, because there is only one broadcaster, when a station sees the start of the broadcast of its packet, it can be assured that the packet's transmission will not fail due to contention. This is not the case, however,

in the CAMB tree. Because of the possibilities of broadcast abortion and preemption, the CAMB tree station protocol requires that a station see the broadcast of the entire packet. (See example in Section 6.4).

The CAMB station protocol is as follows: (This protocol incorporates two modifications, proposed in [7], to the original protocol of [5]).

(1) A station transmits a packet as soon as one is ready, starting at say, time t.

(2) The station monitors the downlink for the broadcast of the packet.

(3) If the station does not see the start of its packet by time $t + R_{pa}$ (where R_{pa} = round trip propagation delay between the station and the proper ancestor)

(4) then it retransmits the packet immediately

(5) else (the station *does* start to see its packet within this time)

  (5-1) if the station sees the broadcast of the packet truncated by the broadcast of another packet then it retransmits the packet immediately

  (5-2) else the station sees the broadcast of the whole packet (and the transmission was successful).

In this protocol the station times out at time $t + R_{pa}$ if it has not seen the start of its packet. This means that each station must keep a table of round trip delays for its proper ancestors. This information could be provided when the network is initialized. Alternatively, at network initialization a station can assume that each table value equals R, the worst case round trip delay between any station and the root. The station can subsequently learn an accurate value for each R_{pa} entry of its table by timing the round trip delay for a message that has that proper ancestor. (This timed message could be a special control message). This approach of timing packets is feasible because, for a packet that is transmitted successfully, there are no non-deterministic delays introduced by the network.

The simplest approach would be to assume $R_{pa} = R$ for every packet transmission in the network, and not worry about variations in round trip delays to different proper ancestors. In this case, no table of delay information need be maintained by the station, and R could be a hardware setting of the station's interface hardware.

6.3. The Effects of Climbing Packets

The switch protocol is designed such that, when a packet is selected by the US of its proper ancestor, in addition to being passed to the DS, the packet is also transmitted on the parent uplink. The following is the motivation for having a packet climb the tree above its proper ancestor:

Every time the packet succeeds in busying the US of a switch above its proper ancestor, it prevents stations beneath that switch from using that switch as a broadcast point. This makes the packet's own broadcast less vulnerable to abortion. If the packet can busy the US of every switch up to and including the root switch, then the packet's broadcast is completely shielded from abortion. To the extent that the packet is unsuccessful in winning switches

above its proper ancestor, the packet simply "takes its chances" and hopes that it won't be aborted.

More generally, the effect of a packet climbing the tree is the partitioning of the tree into broadcast domains. Every time a packet busies the US of a switch above, below, or at its proper ancestor, it defines partitions that are the children subtrees of the switch. Within each of these partitions, broadcasts may occur; but while they proceed they are vulnerable to abortion by broadcasts originating higher in the tree. Any attempt to transmit from within a partition to a destination outside the partition does not succeed. This is illustrated in Figure 5. Each transmission indicated is possible except for the one marked with the cross. It fails because the parent of its partition is busy.

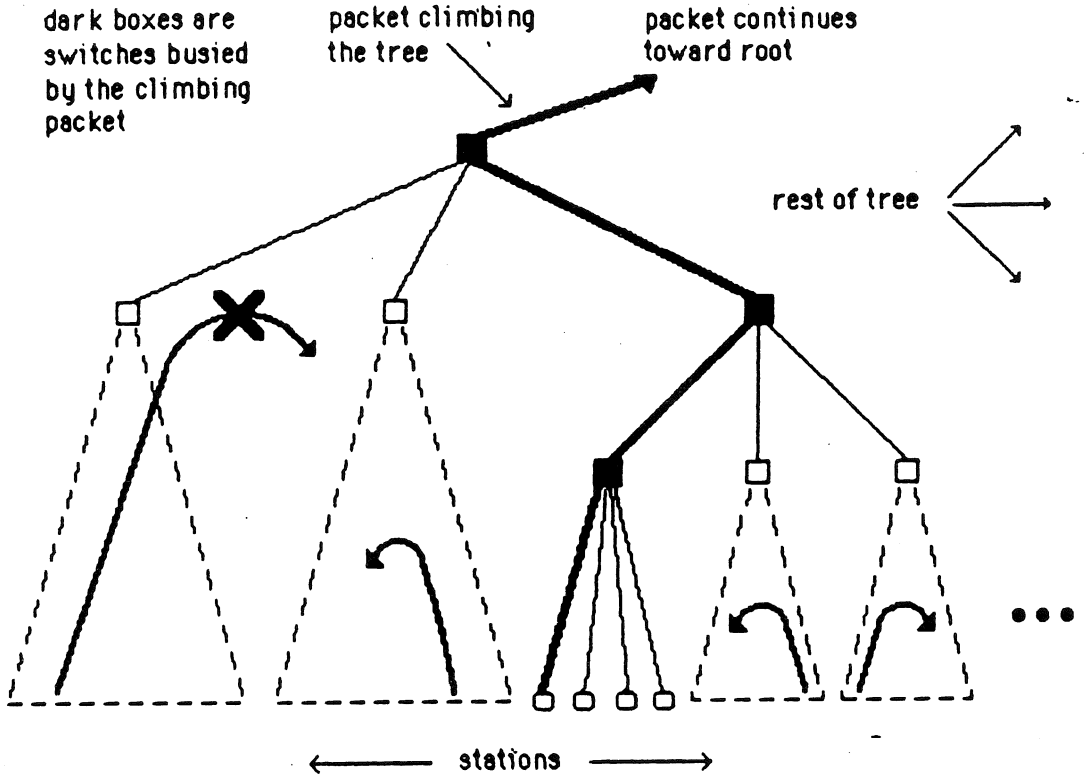

Fig.5 Partitioning of CAMB Tree by Climbing Packet

6.4. Example of Transmissions in a CAMB Tree

Figure 6 illustrates the transmission of packets in a CAMB tree. In this example :

- Station 6 has recently finished transmitting a packet to station 8. 6's packet is currently busying the selector of root the switch, G. Because 6 won the root switch, 6's broadcast is invulnerable to abortion.
- 1 is sending a packet to 3. Because 6's packet had already won the root by the time 1's packet arrived there, 1's broadcast is vulnerable to abortion by a broadcast from the

root.

- 3 wants to send to 6, but 3's packet is blocked at switch E by 1's packet. 3 will to have retransmit its packet.
- 5 tries to send to 6. 5's packet is selected by switch C and is passed by the AR of switch C, but the broadcast of 5's packet has been preempted by the broadcast of 6's packet. 5 will have to retransmit its packet.
- 7 has started to send to 4. 7's packet will get to switch F before 5's packet does. After winning switch F, 7's packet will win the root switch that will soon be vacated by 6's packet. The root is the proper ancestor of 7's packet, and the broadcast of 7's packet will travel down the tree and abort the broadcast of 1's packet at switch E.

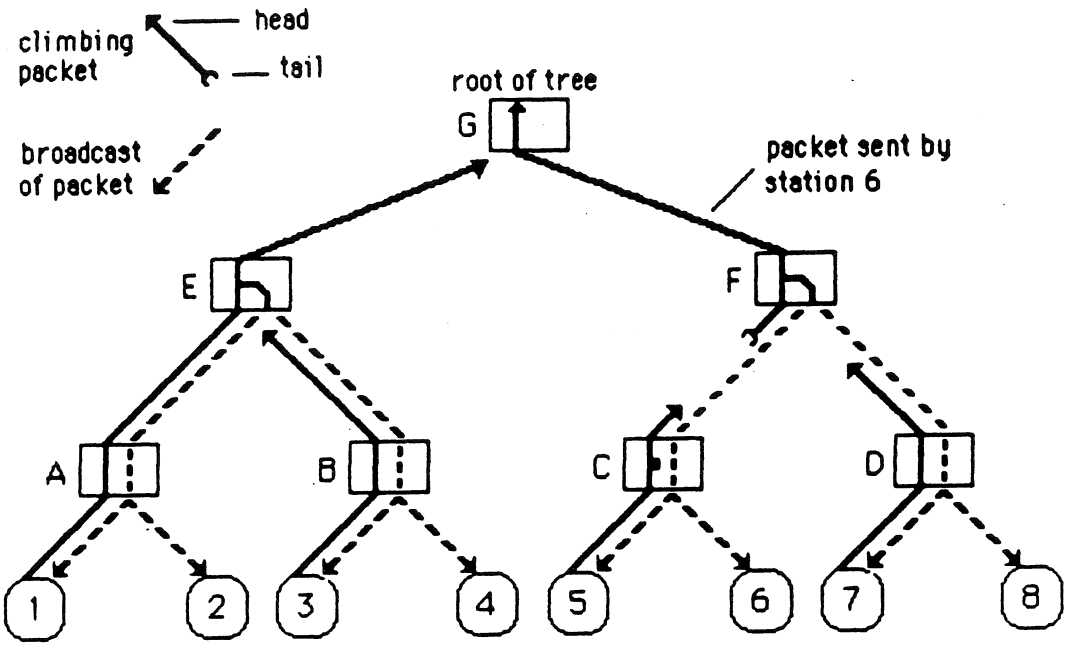

Fig.6 Transmissions in CAMB Tree

The abortion and preemption of broadcasts in this example illustrate the need for a station to check for the broadcast of all of its packet. Station 1 in the example is a case in point. Additionally, if channel speeds are high, packet durations may be small compared to propagation delays. Under this circumstance, it is possible for a station to monitor its downlink and see the broadcast of other packets arrive before seeing the broadcast of its own packet. Thus, even though a station sees the start of another packet, it should wait until the time $t + R_{pa}$ before giving up on seeing its packet.

6.5. Concurrency of Packet Transmissions

A feature of the CAMB tree that is not shared by CSMA/CD networks is the possibility

of supporting concurrent transmissions. Concurrency is of course made possible by the point-to-point, segmented nature of the network, which allows broadcasting to subsets of the tree. This concurrency is paid for by the necessity of each station knowing the map between destination stations and proper ancestor switches, but this seems a reasonable price to pay. We will be running simulations in the near future to determine average concurrencies in a CAMB tree under various traffic conditions.

6.6. Alternate CAMB Protocols

Alternate CAMB switch protocols may give better performance under certain conditions. For instance, in an optical CAMB where channel speeds are high, packets may be small compared to propagation delays, and whole packets may exist in transit on the network. In this case, it may be better not to send packets above their proper ancestor. Sending packet "X" above its proper ancestor might result in the blocking of packets that would not have aborted the broadcast of packet X.

Slightly more complicated switch protocols, which try to take advantage of the deterministic propagation times in the network, are also under consideration. For instance, a slightly more "intelligent" switch can decide if sending a packet any higher above its proper ancestor will further serve to shield it from abortion. To do this, one must add the following fields to the header of each packet:

(1) A flag set by the proper ancestor of the packet which indicates that a packet is above its proper ancestor.

(2) A field set by the source station that indicates the duration of the packet.

(3) A field that contains the accumulated propagation delay of the packet above its proper ancestor. Each switch above the proper ancestor increments this field by an amount equal to the propagation delay from itself to its parent. This delay is an additional piece of knowledge that each switch must have.

By looking at field (1) of a selected packet, the AR knows whether the packet is climbing above its proper ancestor. By comparing the values in fields (2) and (3) of such a packet, the AR of a switch can decide whether a broadcast, that started from the switch's DS at the same time the selected packet arrived, could abort the selected packet at its proper ancestor. This requires adding some simple logic to the AR. If such a broadcast could not abort the packet, then the AR does not pass the packet to the parent uplink, because there is no point in the packet trying to shield against an abortion that could not happen. However, if such a broadcast could abort the packet, the AR updates field (3) and passes the packet to its parent.

The AR can make more sophisticated decisions by adding a simple resetable timer to the switch, which is used to remember how long ago the most recent broadcast by the DS started. For instance, if the AR is transmitting a selected packet on the parent uplink when the DS starts a broadcast, the AR can decide whether that broadcast will abort the selected packet at its proper ancestor below. If the broadcast will abort it, then the AR stops transmitting the packet to the parent uplink.

By making such decisions, there is a reduction in the transmission of aborted packets, which uselessly busy the US's in the network. This should result in an improvement of available concurrency and overall throughput in the network. We intend to present the details of such switch protocols and the simulation of their performance in a future paper.

The CAMB station protocol we have described requires that a station succeed in the transmission of one packet before it attempts the transmission of the next packet in its queue. Other station protocols that allow a station to transmit subsequent packets before knowing the success or failure of prior transmissions, and which uses resequencing of the packets at the destination station may be feasible. The utilization of the network under the described protocol is sensitive to the rate at which stations can retry the transmission. This is particularly of interest as the channel speed increases and the ratio of packet duration to R_{pa} decreases. With stations waiting to learn the fate of such short packets, the idle time of links will tend to become large. The use of a station protocol which allows a station to transmit subsequent packets before knowing the success or failure of prior transmissions would help to increase link utilizations.

The aforementioned protocol modifications may be a good answer for future regimes of optical communication in which channel speeds are high and the ratio of packet length to propagation delay is small. On the other hand, the CAMB tree requires that packets *win* complete paths from source to destination for successful transmission to occur. Whenever a packet fails due to contention, it loses the investment it has in its partially established path. Clearly, as the ratio of the length of the complete transmission path to the length of the packet increases, the loss of partially established paths becomes more serious. In such a regime, it may be better to buffer whole packets within switches and use a store-and-forward protocol. Deciding which approach for LANs is better: simple switches plus complete path retrys, or packet-buffering switches, involves not only questions of performance but also questions of cost. As new technology develops and proliferates, the answers to such questions may change considerably. We consider this area an interesting and fruitful one to research.

7. A Non-Broadcast Tree LAN with Collision Avoidance: Tinker-Tree

Yemini [8] has proposed the Tinker-Tree LAN which uses collision avoidance and random access. The non-leaf nodes are switches and the leaves of the tree are the stations. Switches are connected to each other using full duplex connections such as uplink-downlink pairs like in the CAMB tree. See Figure 7. With a tree topology there is only one path between any pair of source and destination stations. The switches in Tinker-Tree route packets along this unique path using address information contained in packet headers. Tinker-Tree allows for concurrent packet transmissions to the extent that transmission paths do not overlap. Contention along overlapping path segments is resolved by the collision avoidance circuitry in the switches. When two or more packets try to use the same edge in the tree, one packet is allowed access and the other packets are blocked. Packets that are blocked must be retransmitted by their source stations.

Yemini does not propose a specific switch architecture in [8]. Two possibilities are shown in Figure 8. Figure 8(a) shows a switch closely related to a CAMB switch. The Uplink

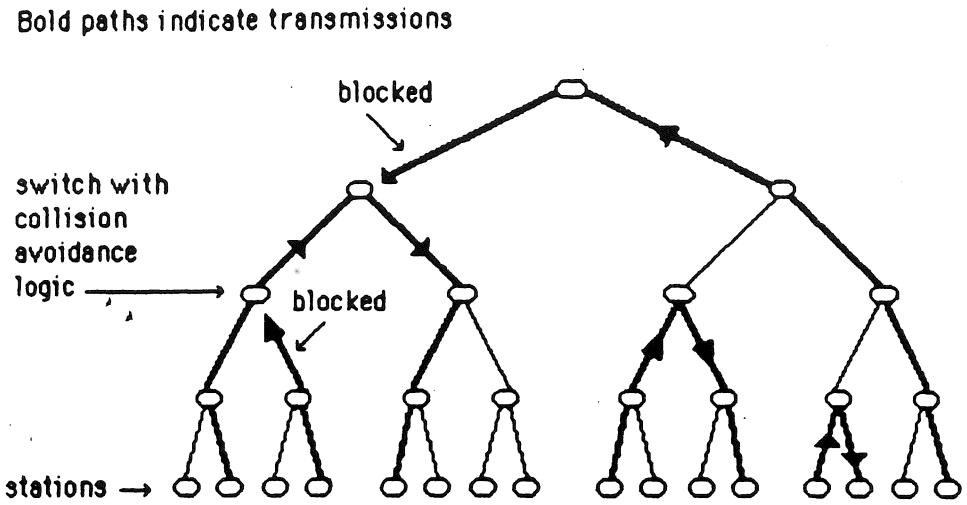

Fig.7 Tinker-Tree Network

Selector is identical. The Address Recognizer is now a Router that routes selected packets to only one of either the parent uplink or the Downlink Selector. The Downlink Selector no longer gives priority to packets from the parent downlink. Instead it functions exactly like the Uplink Selector, and thus implements collision avoidance among its two inputs. The Downlink Selector sends selected packets to a Router identical to the other one. The Routers switch packets according to addressing information in the packet headers.

The switch in Figure 8(a) is capable of supporting at most two simultaneous transmissions, as shown in Figure 8(b). An alternative switch architecture can take advantage of greater concurrency in a Tinker-Tree. This switch uses an $N \times 2$ switch for the uplink side of the switch and a $2 \times N$ switch on the downlink side, where N is the number of children of the switch. The address information in the packet headers can be used to implement self-routing of packets through these switching components. Figure 8(c) shows an architecture with two 2×2 switching components, and Figure 8(d) indicates a state of this switch in which three concurrent transmissions are taking place. The switching components could also support broadcast modes to implement broadcast transmissions in the network. Figure 8(e) indicates the state of the switch for a proper ancestor broadcasting to its subtree, while concurrently, a different transmission continues to climb the tree.

Although packet transmissions may be blocked in Tinker-Tree, they are not aborted. Therefore, when a station starts to see a packet arrive it will see the whole packet arrive. Although Yemini does not address the issue of how a Tinker-Tree station learns of its packet being blocked, one method that could be used is to have the destination station send a short acknowledge packet to the source station. The source station then learns of contention failure by timing out on the receipt of an acknowledgment. This acknowledgement could be generated at a higher layer in the network.

7.1. Comparison of Tinker-Tree and CAMB tree

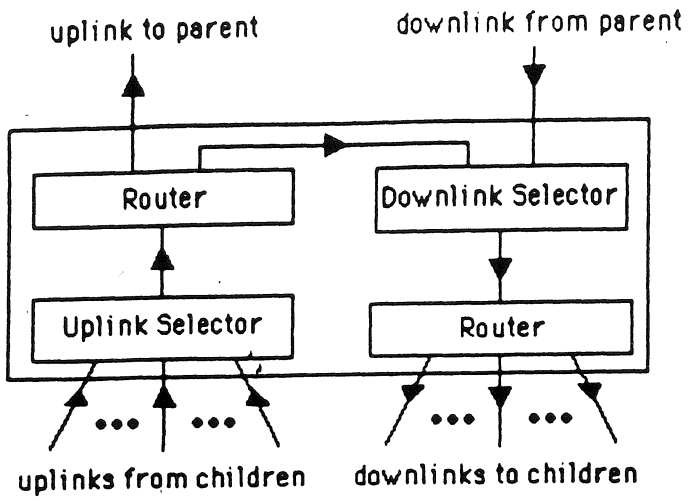

Fig. 8(a)

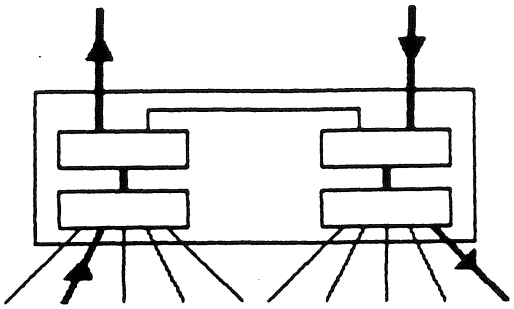

Fig. 8(b)

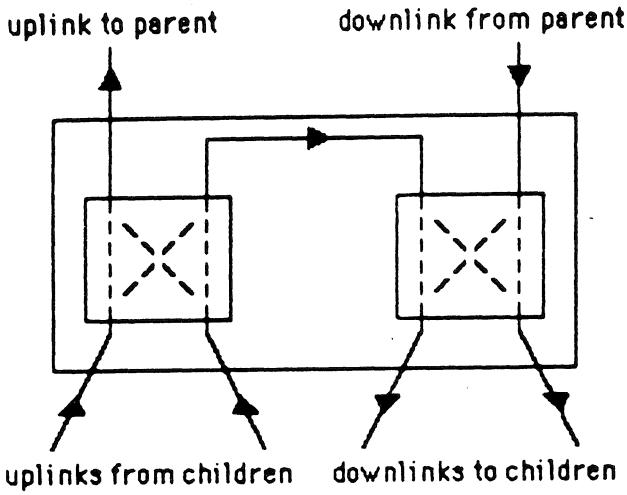

Fig. 8(c)

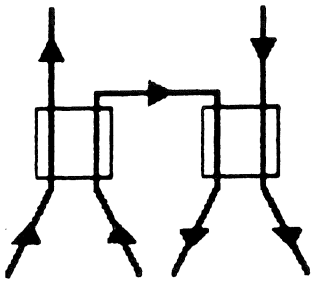

Fig. 8(d)

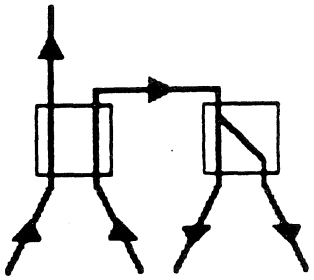

Fig. 8(e)

Fig.8 Possible Tinker-Tree Switches

In both the CAMB tree and Tinker-Tree, stations use time out periods to detect contention based failure. CAMB stations use shorter time outs and get faster feedback on the success or failure of their transmissions because the feedback information travels a shorter distance than it does in the Tinker-Tree. In the CAMB tree, the station waits for its packet to travel to the proper ancestor and back. In Tinker-Tree, the feedback information arrives after two trips between the source and destination stations - one trip for the data packet and another trip for the acknowledgement. Additionally, in Tinker-Tree there may be some queueing delay at the destination station before the acknowledgement is sent, and this delay must be included in the time out period.

On the other hand, the discreteness of routing in the Tinker-Tree allows greater concurrency than in the CAMB tree, which supports concurrent transmissions only in non-overlapping subtrees. Essentially, CAMB uses broadcasting to get faster feedback and thus faster retry rates, but obtains these at the expense of concurrency.

7.2. Combination of CAMB Tree and Tinker-Tree

A third possible tree LAN combines the strengths of the CAMB tree and the Tinker-Tree. This LAN is like the CAMB tree except that the proper ancestor of a packet does not broadcast the packet to its entire subtree. Instead, the proper ancestor routes a copy of the packet on the downlink that leads to the destination station, and it also routes a copy of the packet on the downlink that leads to the source station. Switches below the proper ancestor route these packets to a single appropriate downlink. For convenience, we refer to the transmission by the proper ancestor to both the source and destination stations as *limited broadcast*, and we call this LAN a CAMLB tree (L for "limited").

As in the CAMB tree, a packet that arrives on the parent downlink of a switch is allowed to abort an ongoing limited-broadcast by that switch. Again, this allows the source station to interpret the receipt of its packet as an indication that the packet was also transmitted to the destination station. However, if we use a switch that has a $2 \times N$ switch on its downlink side, then two downlink transmissions can occur concurrently. In this case, broadcast abortion need occur only when the two broadcasts contend for the same child downlink. Similarly, a packet that is selected by it proper ancestor is preempted only if there is contention for the same child downlink. Dual uplink transmissions can also be supported by using an $N \times 2$ switch for packet selection.

The CAMLB station protocol is the same as that used in the CAMB tree: if the source station's packet is blocked while climbing to its proper ancestor, the station times out on seeing its packet on its downlink and retransmits the packet. If the limited broadcast of a packet is aborted, the source station see the packet truncation, and retransmits the packet.

This LAN uses discrete routing like Tinker-Tree, but also provides fast feedback to the source station. This quick feedback allows the source station to retry transmissions quickly, which should result in good utilization of the network's capacity. On the other hand, this feedback does not consume the downlink side of a whole subtree, thus providing for greater concurrency of transmissions in the tree.

8. Performance Analysis of a Broadcast Star Network with Infinite Station Population

8.1. Transmission Delay

In this section we analyze the performance of a broadcast star network and obtain distribution of the transmission delay of packets. Although a broadcast star network does not require synchronous operation, we assume in the analysis that the time is slotted and is measured by slots. Stations transmit only at the beginning of a slot. The length of a packet is assumed to be constant and its transmission time is equal to the slot length. Newly arrived packets are transmitted in the slot next to their arrival instants. More than one stations

may transmit packets in the same slot. In this case, we assume that the switch chooses one packet randomly and broadcasts it on the downlinks. The other packets are blocked at the switch. Upon the reception of a broadcast packet, a station knows whether its transmission succeeds. Blocked packets are retransmitted immediately. We assume infinite station population, which collectively generates new packets according to Poisson distribution with rate λ (packets/slot).

We define the total transmission delay D as the time from an arrival of a packet to its successful reception by the destination station. Let τ be the time from the arrival of a packet to the beginning of the next slot, and m be the number of retransmissions required for a packet to be successfully transmitted.

Since it takes $R+1$ for a station to know whether a transmission is a success or not and blocked packets are immediately retransmitted, a retransmission requires $\lceil R \rceil + 1$ slots, where R denotes a propagation time to and from the switch, and $\lceil R \rceil$ denotes the least integer greater than or equal to R. Then D becomes,

$$D = \tau + m \times (\lceil R \rceil + 1) + R + 1. \tag{1}$$

Note that random variables τ and m are independent. From eq.(1), the average and variance of D becomes,

$$E[D] = E[\tau] + E[m](\lceil R \rceil + 1) + R + 1, \tag{2}$$

$$Var[D] = Var[\tau] + Var[m](\lceil R \rceil + 1)^2. \tag{3}$$

Since we assume Poisson arrivals, arrival points are uniformly distributed within a slot. Hence, we have $E[\tau] = \frac{1}{2}$ and $Var[\tau] = \frac{1}{12}$. From eqs.(2) and (3), we have

$$E[D] = \frac{1}{2} + E[m](\lceil R \rceil + 1) + R + 1, \tag{4}$$

$$Var[D] = \frac{1}{12} + Var[m](\lceil R \rceil + 1)^2. \tag{5}$$

Higher moments of D can be also obtained from eq.(1).

In the next section, we obtain the steady state distribution of the number of the retransmissions m.

8.2. Conditional Moment of the Number of Retransmissions

Let $P_N(m)$ be the conditional probability that a packet (say, test packet) requires exactly m retransmissions to be successfully transmitted given that N packets (including the test packet itself) access in the same slot.

The probability that the test packet is successfully transmitted at the first trial is $\frac{1}{N}$. Thus, we have

$$P_N(0) = \frac{1}{N} \tag{6}$$

If the test packet is blocked at a switch, which happens with the probability $1 - \frac{1}{N}$, it is retransmitted $\lceil R \rceil$ slots later. If we let k be the number of new arrivals in the slot immediately prior to the retransmission, the probability of a successful retransmission is $\frac{1}{N+k-1}$, since $N-1$ blokced packets and k new packets access in that slot. Thus, we have the following recursive equation.

$$P_N(m) = (1 - \frac{1}{N}) \sum_{k=0}^{\infty} \frac{\lambda^k}{k!} e^{-\lambda} P_{N+k-1}(m-1) \qquad (m \geq 1) \tag{7}$$

Note that we assume a Poisson process for new packet arrivals.

Let M_N^l be the l-th conditional moment of the number m of retransmissions given that N packets access in the same slot, namely,

$$M_N^l = \sum_{m=0}^{\infty} m^l P_N(m) \tag{8}$$

From the definition, we have $M_N^0 = 1$. From eq.(7), we have

$$\begin{aligned}
M_N^l &= P_N(0)0^l + \sum_{m=1}^{\infty} P_N(m) m^l \\
&= \sum_{m=1}^{\infty} (1 - \frac{1}{N}) \sum_{k=0}^{\infty} \frac{\lambda^k}{k!} e^{-\lambda} P_{N+k-1}(m-1) m^l \\
&= (1 - \frac{1}{N}) \sum_{k=0}^{\infty} \frac{\lambda^k}{k!} e^{-\lambda} \sum_{m=1}^{\infty} ((m-1)+1)^l P_{N+k-1}(m-1) \\
&= (1 - \frac{1}{N}) \sum_{k=0}^{\infty} \frac{\lambda^k}{k!} e^{-\lambda} \sum_{m=1}^{\infty} \sum_{r=0}^{l} \binom{l}{r} (m-1)^r P_{N+k-1}(m-1) \\
&= (1 - \frac{1}{N}) \sum_{k=0}^{\infty} \frac{\lambda^k}{k!} e^{-\lambda} \sum_{r=0}^{l} \binom{l}{r} M_{N+k-1}^r
\end{aligned} \tag{9}$$

Kingman [9] analyzed the waiting time distribution of random-service continuous-time M/G/1 queueing system. (Our model for a broadcast star network is random service M/D/1, but with slotted time.) He has proved that the equation of the same type as eq.(9) has a unique solution, which is of a polynomial of N of order l. Following Kingman's approach, we assume the following polynomial as a solution to eq.(9).

$$M_N^l = \sum_{k=0}^{l} a_k^l N^k \tag{10}$$

where a_k^ls are coefficients of the polynomial.

By substituting $M_N^1 = a_0^1 + a_1^1 N$ and $M_N^2 = a_0^2 + a_1^2 N + a_2^2 N^2$ to eq.(9), we determine the coefficients. Thus, we have

$$M_N^1 = \frac{N-1}{2-\lambda} \tag{11}$$

$$M_N^2 = \frac{2(N^2-3N+2)}{(2-\lambda)(3-2\lambda)} + \frac{(6-\lambda)(N-1)}{(2-\lambda)^2(3-2\lambda)} \tag{12}$$

By removing a condition on N, we have

$$E[m] = \frac{1}{2-\lambda}(E[N]-1) \tag{13}$$

$$E[m^2] = \frac{2(E[N^2]-3E[N]+2)}{(2-\lambda)(3-2\lambda)} + \frac{(6-\lambda)(E[N]-1)}{(2-\lambda)^2(3-2\lambda)} \tag{14}$$

In the next subsection, we will obtain the distribution of N, and its first and second moments ($E[N]$ and $E[N^2]$).

8.3. Distribution of the Number of Packets in the System

We observe the system at the beginning (more precisely, immediately after the beginning) of slots where transmission of a packet (say, test packet) takes place. See Fig.9. The circles in Fig.9 denote these observation time points, or imbedded points.

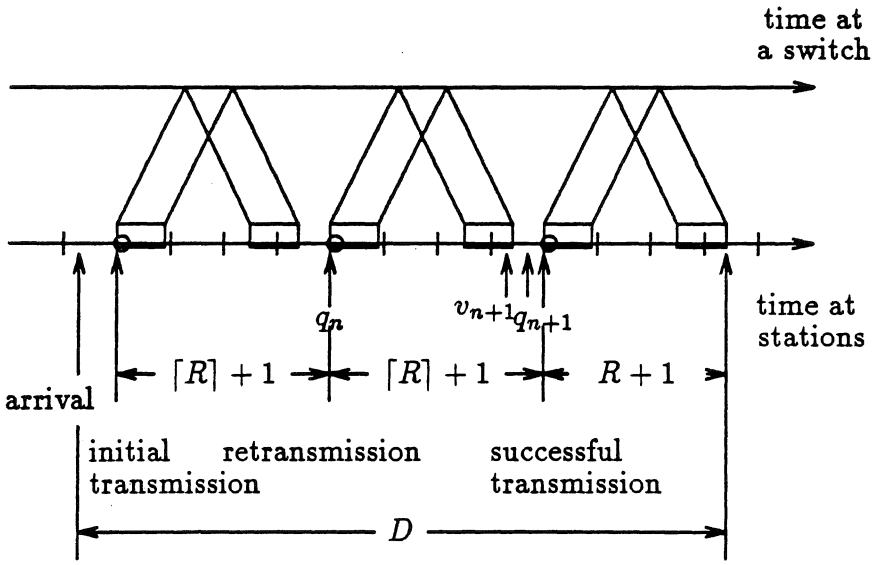

Fig.9. Imbedded Points

Let q_n be the number of the blocked packets at the n-th imbedded point. In other words, $q_n + 1$ is the number of packets accessing the channel at the n-th imbedded time point, and one of these $q_n + 1$ packets is successfully transmitted. Further, let v_{n+1} be the number of

new packet arrivals in the slot immediately prior to the $(n+1)$-st imbedded point. See Fig.9. Then q_{n+1} becomes

$$q_{n+1} = \begin{cases} q_n + v_{n+1} - 1 & (q_n > 0) \\ max\{v_{n+1} - 1, 0\} & (q_n = 0) \end{cases} \tag{15}$$

Let $Q_n(z)$ and $V_n(z)$ be the z-transforms for q_n and v_n, respectively. That is,

$$Q_n(z) = \sum_{i=0}^{\infty} Prob[q_n = i] z^i \tag{16}$$

$$V_n(z) = \sum_{i=0}^{\infty} Prob[v_n = i] z^i \tag{17}$$

We assume the steady state exists for q_n, namely,

$$\lim_{n \to \infty} q_n = q \tag{18}$$

$$Q(z) = \lim_{n \to \infty} Q_n(z) = \sum_{i=0}^{\infty} Prob[q = i] z^i \tag{19}$$

Since we assume Poisson arrivals, steady state distribution for $v = \lim_{n \to \infty} v_n$ exists and is given by

$$Prob[v = i] = \frac{\lambda^i}{i!} e^{-\lambda} \tag{20}$$

z-transform is given by

$$V(z) = \sum_{i=0}^{\infty} Prob[v = i] z^i = e^{\lambda(z-1)} \tag{21}$$

If $q_n > 0$, q_{n+1} is equal to the number of new arrivals in the slot prior to the $(n+1)$-st imbedded point (v_{n+1}), plus the number of blocked packet at the n-th imbedded point (q_n), minus one (successful transmission at the $(n+1)$-st imbedded point). Refer to eq.(15). Hence, we have

$$Q_{n+1} = \frac{Q_n(z) - Q_n(0)}{z} V_{n+1}(z) \tag{22}$$

If $q_n = 0$, and if there is at least one arrival in the slot prior to the $(n+1)$-st imbedded point, q_{n+1} is equal to the number of new arrivals in the slot prior to the $(n+1)$-st imbedded point (v_{n+1}), minus one (successful transmission at the $(n+1)$-st imbedded point). If $q_n = 0$ and $v_{n+1} = 0$, q_{n+1} is zero. Refer to eq.(15), again. Hence, we have

$$Q_{n+1} = Q_n(0)[V_{n+1}(0) + (\frac{V_{n+1}(z) - V_{n+1}(0)}{z})] \tag{23}$$

From eqs.(22) and (23), we have

$$Q_{n+1}(z) = \frac{(Q_n(z) - Q_n(0))V_{n+1}(z)}{z} + Q_n(0)[V_{n+1}(0) + (\frac{V_{n+1}(z) - V_{n+1}(0)}{z})] \quad (24)$$

By taking a limit where n goes to infinity, we have

$$Q(z) = \frac{(Q(z) - Q(0))V(z)}{z} + Q(0)[V(0) + (\frac{V(z) - V(0)}{z})] \quad (25)$$

By solving the above equation with respect to $Q(z)$, we have

$$Q(z) = \frac{1}{z - V(z)}Q(0)V(0) = \frac{z-1}{z - e^{\lambda(z-1)}}Q(0)V(0) \quad (26)$$

By substituting z=1, using L'hopital's theorem and $Q(1) = V(1) = 1$, we have

$$\lim_{z\to 1} Q(z) = \lim_{z\to 1} \frac{z-1}{z - e^{\lambda(z-1)}}Q(0)V(0) = \frac{\frac{d}{dz}(z-1)}{\frac{d}{dz}(z - e^{\lambda(z-1)})}Q(0)V(0)$$
$$= \frac{Q(0)V(0)}{1-\lambda} = 1 \quad (27)$$

Thus, we have

$$Q(0)V(0) = 1 - \lambda \quad (28)$$

By substituting eq.(28) to eq.(26), finally we have,

$$Q(z) = \frac{(1-\lambda)(z-1)}{z - e^{\lambda(z-1)}} \quad (29)$$

From the distribution of q, we obtain the distribution of the number N of the packets accessing in a slot given that the test packet arrives in the slot immediately prior to that slot.

Let $P(z)$ denote the z-transform of the distribution of N. Since the blocked packets and new packets access in a slot, $P(z)$ is given by the convolution of $Q(z)$ and the conditional z-transform for the distribution of new arrivals in a slot given one or more packets arrive in the slot. Since the probability of having no arrival in a slot is $V(0) = 1 - e^{-\lambda}$, z-transform of the number of new arrivals in a slot given one or more packets arrive in the slot is

$$\frac{V(z) - V(0)}{1 - V(0)} \quad (30)$$

Therefore, we have

$$P(z) = Q(z)\frac{V(z) - V(0)}{1 - V(0)} \quad (31)$$

From eqs.(21), (29) and $V(0) = 1 - e^{-\lambda}$, $P(z)$ becomes

$$P(z) = \frac{(1-\lambda)(z-1)}{z - e^{\lambda(z-1)}} \frac{e^{\lambda(z-1)} - e^{-\lambda}}{1 - e^{-\lambda}} \tag{32}$$

By substituting $z = 1$ in the i-th derivative of $P(z)$, i-th moment of the distribution of N is derived. The first two moments are as follows,

$$P'(1) = E[N] = \frac{-(1+e^{-\lambda})\lambda^2 + 2\lambda}{2(1-\lambda)(1-e^{-\lambda})} \tag{33}$$

$$P''(1) = E[N(N-1)] = E[N^2] - E[N] = \frac{(1-e^{-\lambda})\lambda^4 - (4+2e^{-\lambda})\lambda^3 + 6\lambda^2}{6(1-\lambda)^2(1-e^{-\lambda})} \tag{34}$$

9. Numerical Results

In this section, we show some numerical results for a broadcast star network. Fig.10 shows average total transmission delay, $E[D]$, of a broadcast star network as a function of channel load λ. This figure shows that the average delay for various values of R (round trip propagation delay) has the similar behavior and that delay grows rapidly when channel load exceeds 0.8. Maximum throughput is 1.0; in other words, the network becomes saturated at the load of 1.0. This is intuitively clear, because at least one packet is successfully transmitted per slot in a broadcast star network.

Fig.11 shows the variance of delay as a function of the channel load. The vertical scale is logarithmic. In this figure variance of delay increases slowly until the channel load exceeds 0.8. For the higher load, variance grows rapidly to infinity.

10. Conclusions

In this paper, we proposed a new network architecture based on collision avoidance. We presented various protocol alternatives and switch architectures, along with the performance analysis of the simplest such network called a broadcast star network.

References

[1] J. Kurose, M. Schwartz and Y. Yemini, "Multitple-Access Protocols and Time-Constrained Communication", ACM Computing Surveys, Vol.16, No.1, March 1984.

[2] Felix Closs and Robert P. Lee, "A Multi-Star Broadcast Network for Local-Area Communications", in Local Networks for Computer Communications, A. West and P. Janson eds., North-Holland Publishing Company, cp IFIP, 1981.

[3] A. Albanese, "Star Network with Collision-Avoidance Circuits", The Bell System Technical Journal, Vol. 62, No.3, March 1983.

[4] E. Stewart Lee and Peter I. P. Boulton, "The Principles and Performance of Hubnet: A 50 Mbit/s Glass Fiber Local Area Network", IEEE Journal on Selected Areas in Communications, Vol. SAC-1, No.5, November 1983, pp.711-720.

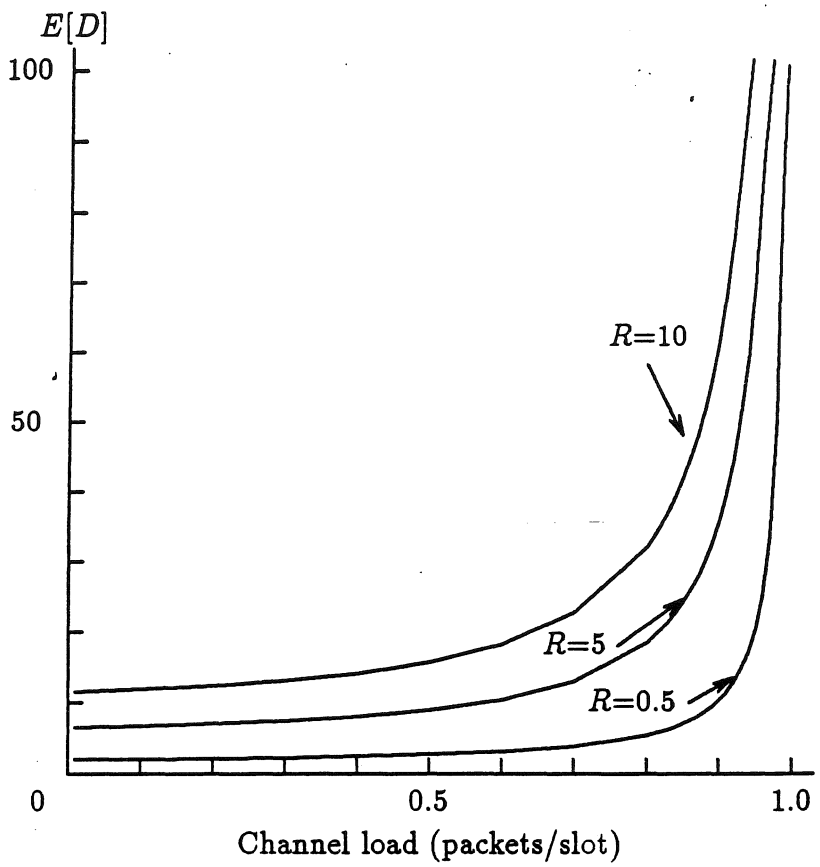

Fig.10 Transmission Delay

[5] Tatsuya Suda, Yechiam Yemini and Mischa Schwartz, "Tree Network with Collision Avoidance Switches", Proc. of Infocom '84, pp.105-113.

[6] A. E. Kamal, "A Performance Model for a Star Network", Proc. of Globecom '86, pp.12-18.

[7] V. Ielapi, S. Marano and A. Volpentesta, "A Simulation Study for a Tree Local Area Network with Concurrent Transmissions", in Local Communications Systems: LAN and PBX, Elsevier Science Publishers BV, cp IFIP, 1987.

[8] Yechiam Yemini, "Tinkernet: or, Is There Life Between LANs and PBXs?", ICC '83.

[9] J.F.C. Kingman, "On Queues in which Customers are Served in Random Order," Proc. Camb. Phil. Soc., Vol.58, pp.79–91(1962).

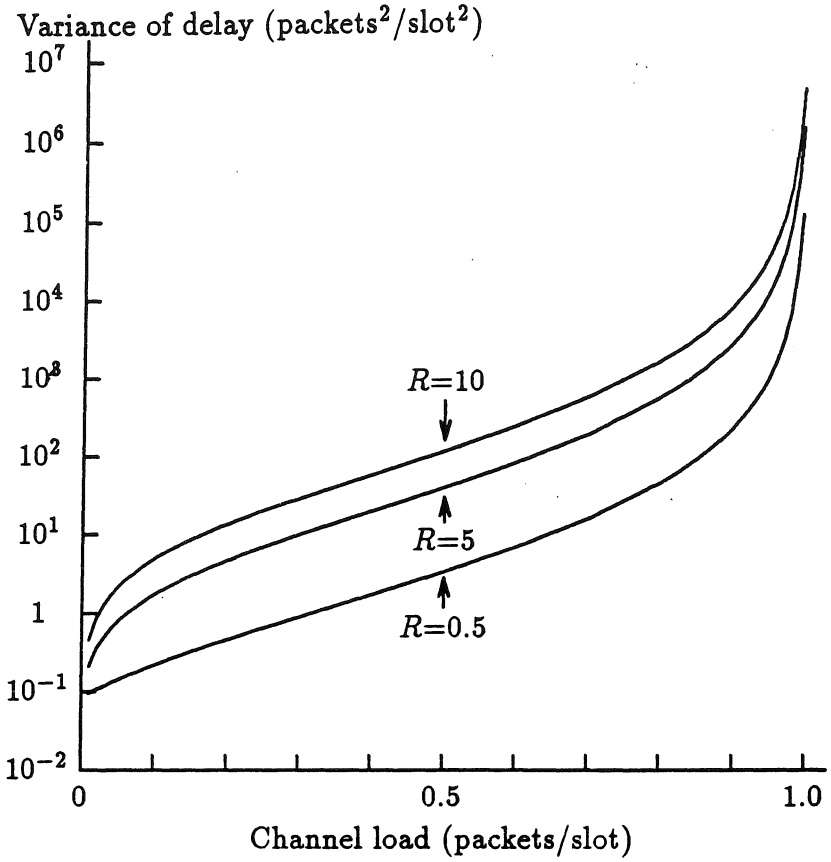

Fig.11 Variance of Delay